Remarks

Entry of the amendments, reconsideration of the application, and allowance of all pending claims are respectfully requested. Claims 1, 3-4, 9-11, 14-18, 21-22, 27, 29-30, 35-37, 40-44, 47-48, 53-56, 58, 60-61, 66-68, and 70-75 & 78-79 remain pending.

By this amendment, independent claims 1, 27, 53 & 58 are amended to more particularly point out and distinctly claim certain features of applicants' invention. These amendments to the claims constitute a bona fide attempt to advance prosecution of this application, and are not meant to acquiesce to the outstanding rejections. Support for these amendments can be found throughout the application. See, for example, p. 14, lines 12-17; p. 19, line 26 – p. 20, line 6; and p. 24, lines 18-21 of applicants' specification, and prior claims 19, 20, 45, 46, 76 & 77 (canceled herein). Thus, no new matter is added to the application by any amendment presented.

In the Office Action, claims 1, 3-4, 9-11, 14-22, 27, 29-30, 35-37, 40-48, 53-56, 58, 60-61, 66-68 & 70-79 were rejected under 35 U.S.C. §103(a) as being unpatentable over Schoening et al. (U.S. Patent No. 6,205,465; hereinafter, "Schoening") in view of Belkin et al. (U.S. Patent No. 6,542,920; hereinafter, "Belkin"). This rejection is respectfully, but most strenuously, traversed to any extent deemed applicable to the independent claims presented herewith.

An "obviousness" determination requires an evaluation of whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art. In evaluating claimed subject matter as a whole, the Federal Circuit has expressly mandated that functional claim language be considered in evaluating a claim relative to the prior art. Applicants respectfully submit that the application of these standards to the independent claims presented herewith leads to the conclusion that the recited subject matter would not have been obvious to one of ordinary skill in the art based on the applied patents.

Applicants' invention is directed to the management of thread pools to service a request from a requester (e.g., a client) and a response (e.g., callback response) from another requester on which the request is waiting. For instance, a client request for a locked data file waits for

another client's callback response that unlocks the data file. This thread pool management technique avoids deadlock between the request and the response on which the request is waiting by dynamically altering an eligible thread pool set (e.g., containing a primary thread pool) to provide an altered set of thread pools. The altered thread pool set includes, for example, the primary thread pool and a secondary thread pool to service the response. By allowing, for example, the secondary thread pool to service the response and providing the primary thread pool to service the request waiting on the response, deadlock is avoided. This deadlock avoidance scheme is efficient because it does not require any input from the requesters that issue the request or the response. The thread pool management technique claimed uses low-level operating system functionality that is incapable of having knowledge of client input.

As one example, applicants claim a technique for managing thread pools of a computing environment (e.g., claim 1). This technique includes receiving from a first requester of the computing environment a request to be processed. This request is waiting on a response from a second requester of the computing environment, and the response is to be serviced by a thread pool selected from a set of one or more eligible thread pools. The technique further includes, upon receipt of the request waiting on the response, and without input from the first requester or the second requester of which thread pools can service the response, dynamically altering the set of one or more eligible thread pools to provide an altered thread pool set of eligible thread pools, wherein a thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response. The dynamically altering comprises setting a pool mask to indicate the eligible thread pools of the altered thread pool set to service the response. Applicants respectfully submit that at least the feature of dynamically altering the set of one or more eligible thread pools without input from the first requester or second requester, and wherein a thread pool of the altered thread pool set is to service the response to avoid deadlock, is not taught, suggested or implied by Schoening and Belkin, alone or in combination.

Schoening describes determining whether or not work can be run in parallel in a multi-threaded environment. When parallel processing is used, Schoening dispatches the work onto separate threads based on a partial order determined by preconditions and resource requirements

associated with execution components. (see Abstract and Col. 4, lines 1-29 thereof). This parallel processing technique is different from the protocol of the present invention.

For example, applicants' claimed invention recites, in part, dynamically altering the set of one or more eligible thread pools without input from the first requester or the second requester of which thread pools can service the response. In contrast, Schoening determines, with input, which thread pools can service a response. The input used by the Schoening patent in its determination of thread pools is the partial order (e.g., evaluation sequence), which is "declared or stored in a Partial Order object and is passed as a parameter to the EvalGroup" (col. 41, lines 4-5). As depicted in FIG. 5A of Schoening, parameters associated with the evaluation sequence or partial order are passed to EvalGroup 502 from Service Module Functions (SMFs) 512 (see also col. 4, lines 6-7). SMFs receive such parameters from clients subsequent to clients acquiring access to service modules (see steps 418, 422 & 424 of FIG. 4A thereof). For at least these reasons, applicants respectfully submit that Schoening fails to teach, suggest or imply the above-noted aspect of applicants' invention.

Further, applicants' claim dynamically altering the set of one or more eligible thread pools (without input from the first or second requesters) to provide an altered thread pool set of eligible thread pools, wherein a thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response. In contrast, Schoening does not discuss a deadlock avoidance scheme at all. Instead, Schoening describes the existence of deadlock in the case of synchronizing parallel threads (col. 3, lines 24-30) and the detection of deadlock associated with the transaction processing functions of the Asynchronous Network Interface (ANI). Although the existence and detection of deadlock is disclosed by Schoening, applicants respectfully submit that Schoening does not suggest or imply the avoidance of deadlock with a request awaiting a response using a protocol as claimed in the present invention.

In support of the rejection of the prior independent claims, the Office Action stated that Schoening teaches altering thread pools and cited col. 40, lines 61-62; col. 41, lines 3-14 & 39-42; and col. 42, lines 22-24. These sections of Schoening describe using transactions to synchronize threads; passing a partial order as a parameter to EvalGroup and evaluating that

partial order; processing an execution manager subsystem object (EvalGroup) in parallel with other objects; and providing the partial order of SMFs. To the extent these cited sections are deemed applicable to the claims presented herewith, applicants traverse any conclusion that they teach, suggest or imply the above-noted features of applicants' claimed invention. As noted, the passing of the partial order parameter (e.g., the evaluation sequence) indicates a thread pool determination using input from requesters of which thread pool can service a request, which differs from applicants' recited dynamic altering of the set of one or more eligible thread pools without input from the first or second requesters of which thread pools can service the response.

To summarize, Schoening fails to teach, suggest or imply at least the above-described recited features of (1) dynamically altering the set of one or more eligible thread pools without input from the first requester or second requester of which thread pools can service the response; and (2) the dynamically altering providing an altered thread pool set, wherein a thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response. Applicants respectfully submit that Belkin does not overcome these deficiencies of Schoening as applied against the claims of the present invention.

For example, Belkin determines which thread pool a request is to be associated with input from a request (i.e., from a requester). In particular, Belkin describes in the Abstract thereof:

> . . . When a request is received, it is processed to determine with which thread pool the request is to be associated. This processing is carried out by determining the type of service being requested by the request, and then determining which thread pool is associated with that type of service. Alternatively, this processing is carried out by extracting a set of indication information (e.g., a universal resource identifier) from the request, and then determining which thread pool is associated with that set of indication information . . .

Belkin also discloses that the determination of the thread pool utilizes user-defined tables. For example, Belkin states that "with tables 200 and 300a, the user can freely define any thread pool, and associate any type of service with any defined thread pool" (col. 7, lines 44-46; see also FIGs. 2 & 3 thereof). Moreover, relative to user-defined table 200, Belkin describes an evaluation function to determine which thread pools are appropriate to service a request (col. 15, lines 29-44). This evaluation function's determination is "user-provided" and "based upon the

request" (col. 15, lines 32-36). Thus, the determination in Belkin of which thread pool to use is performed with input from a user of which thread pool is to service a request, which is clearly different from a process which dynamically alters a set of one or more eligible thread pools without input from a first requester or a second requester of which thread pools can service the response, as recited in the claims presented herewith.

Further, Belkin describes detection of deadlock when a thread pool has no free threads available (col. 16, lines 3-5). Belkin's detection of deadlock is directed to invoking the evaluation function to order requests on a queue associated with the thread pool (col. 16, lines 14-43). The invocation of the evaluation function in Belkin does not address deadlock avoidance, per se, as claimed by the present invention. Instead, it provides the dispatching code in Belkin with input regarding the thread pool on which the request is to be placed, and where to put the request on the queue. Moreover, this deadlock-prompted action associated with input of which thread pool can service a request is different from the deadlock avoidance recited by the claims presented herewith, wherein a thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response, and the altered thread pool set is provided by the dynamic alteration of the set of one or more eligible thread pools without input from the first requester or the second requester of which thread pools can service the response.

In the Office Action, it is stated that Belkin teaches dynamically determining different eligible thread groups, and col. 4, lines 38-39; col. 7, lines 41-42; col. 15, lines 33-35 and col. 16, lines 44-46 of Belkin are cited. These sections disclose implementing multiple thread pools; specifying, by means of a row of a table, an association between a particular thread pool and a particular type of service; invoking the user-provided evaluation function to evaluate a request; and invoking the evaluation function by a request processing mechanism. Applicants respectfully traverse any conclusion that these cited sections of Belkin teach, suggest or imply the subject matter of the claims presented herewith. As noted, the invocation of the evaluation function based on a request and the user-defined associations in the table provide input of which thread pools can service a request, which is in contrast to the dynamic alteration without input

from the first or second requesters of which thread pools can service a response, as recited in the claims presented herewith.

Since both Schoening and Belkin fail to teach or suggest applicants' claimed protocol of (1) dynamically altering the set of one or more eligible thread pools without input from the first requester or second requester of which thread pools can service the response; and (2) the dynamically altering providing an altered thread pool set, wherein a thread pool of the altered thread pool set is to service the response to avoid a deadlock with the request awaiting the response, it is respectfully submitted that the combination does not render the present invention obvious.

For the above reasons, applicants respectfully request reconsideration and withdrawal of the obviousness rejection of independent claims 1, 27, 53 & 58. The dependent claims are believed patentable for the same reasons as the independent claims from which they directly or ultimately depend, as well as for their own additional characterizations.

All claims are believed to be in condition for allowance and such action is respectfully requested.

Should the Examiner wish to discuss this case further with applicants' attorney, the Examiner is invited to telephone their below-listed representative.

Respectfully submitted,

Kevin P. Radigan

Kevin P. Radigan
Attorney for Applicants
Registration No. 31,789

Dated: February 19, 2004

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203
Telephone: (518) 452-5600
Facsimile: (518) 452-5579